Fixed-Point Neural Network Ensembles for Visual Navigation

Mauricio A Dias, Fernando S Osorio
Mobile Robotics Lab (LRM)
Institute of Mathematical Science and Computation (ICMC)
University of Sao Paulo (USP)
Sao Carlos - SP - Brazil
macdias,fosorio@icmc.usp.br

Abstract— **Visual navigation is an important research field in robotics because of the low cost of cameras and the high performance that is usually achieved by visual navigation systems. Pixel classification as a road pixel or a non-road pixel is a task that can be well performed by Artificial Neural Networks. In the case of real-time instances of the image classification problem, as when applied to autonomous vehicles navigation, it is interesting to achieve the best possible execution time. Hardware implementations of these systems can achieve fast execution times but the floating-point implementation of Neural Networks are commonly complex and resource-intensive. This work presents the implementation and analysis of a fixed-point Neural Network Ensemble for image classification. The system is composed by six fixed-point Neural Networks verified with cross-validation technique, using some proposed voting schemes and analyzed considering the execution time, precision, memory consumption and accuracy for hardware implementation. The results show that the fixed-point implementation is faster, consumes less memory and has an acceptable precision compared to the floating-point implementation. This fact suggests that the fixed point implementation should be used in systems that need a fast execution time. Some questions about ensembles and voting have to be reviewed for fixed-point Neural Network Ensembles.**

I. INTRODUCTION

Visual navigation is a important and largely used method to do robotic navigation. There are important works in robotics that use visual information for navigation as the cars that participated of DARPA Grand Challenge and DARPA Urban Challenge [1][2]. Considering robotic platforms that are being researched recently, a considerable number of them are critical embedded systems as autonomous vehicles, autonomous airplanes, autonomous robots that dismantle bombs and save buried people. These robots need navigation systems that work in real-time, which in this case means, that give an accurate response in an acceptable execution time.

There are many ways to use visual information for robotic navigation. One way to do it is classifying which pixel (or group of pixels) of an image represents regions where the robot can navigate without problems, and which represents regions that the robot cannot use for navigation. Fast response time and generalization are important features for a classifier that will be included in a visual navigation system and this work's chosen classifiers are the Artificial Neural Networks.

Artificial Neural Networks (ANNs) are computational structures inspired by human brain. These structures, as the human brain, are able to do pattern recognition, that means, receive a pattern signal and assign it to one of a prescribed number of classes [3]. ANNs perform pattern recognition after a training phase. A group of training patterns, composed by inputs and their desired output, is presented to the network repeatedly and by a learning process. This learning process modifies network's weights and the structure learns about the problem. After that, a new pattern is presented to the network and the structure is able to classify that pattern with an accuracy rate higher than before. Some tasks are complex and the final recognition result can be increased by using a group of networks to classify the same data instead of only one network. The groups of Neural Networks are called Neural Network Ensembles (NNEs) [4]. Running a group of networks is a device borrowed from fault-tolerant computing and is based on the fact that each network will make generalization errors on different subsets of the input space lowering the probability of recognition error compared to a single network [4]. There are many ways to gather the results and turn them into a final result, each one recommended to be used for a different situation.

After training, the response time of the networks is almost instantaneous and this feature is important for visual navigation systems. An alternative to decrease even more the response time is to implement the ANNs or NNEs in hardware, also known as Hardware Neural Networks (HNNs) [5]. Usually ANNs use floating-point arithmetic for training and for execution but the hardware implementation of floating-point operations is very expensive considering resources consumption and execution time. The alternative in this case is to implement the network using fixed-point arithmetic. This alternative will result in a final hardware that needs a lower chip area but the accuracy of the result depends on how the fixed-point arithmetic is implemented and the number of representation bits.

Hardware implementation of ANNs can be faster and consume less area and power compared to networks that run on general purpose computers. This fact, together with the natural parallelism of Neural Networks, justifies the research area of HNN. Despite of it, general purpose computers are getting more powerful and software development is being done without thinking about hardware because it is easier and faster. The related works presented here are about HHNs and ANN for robotics separately, it is very difficult to find works about HNN implemented for robotic applications and even more difficult to find fixed-point networks for this purpose with a deep analysis of the fixed-point arithmetic usage impact.

The work presented in [6] implemented a fixed-point neural network to shape recognition. This network was applied on a robotic arm system and the problem is that the fixed-point implementation was not well described. The results showed that hardware implementation was fast but was not really compared to other results. Some works present fixed-point neural network implementations in hardware and software [7][8][9]. These works present good results but they only consider the network implementation, or the application is not directly related to robotics. Neural Networks are widely used in robotics [10], and have many hardware implementations [5] but it is difficult to find works that put together hardware Neural Networks and robotic applications.

This work implements a visual navigation system the uses ANNS and NNEs as classifiers based on previous work [11]. Cited previous work found some relevant features of an image for its pixels classification, proposed ANN architectures and parameters, and used NNEs but didn't use the recommended consensus scheme to decide the collective classification by vote [12][13]. This work implements ANNs and ENNs considering the same image features but exploring different architectures, parameters and voting schemes for the NNE results, and also implements the same ANNs and ENNs in fixed-point arithmetic to evaluate the precision, execution time, memory consumption and accuracy. The final result will be consider for the future hardware implementation of the system.

II. METHOD

This work's development method works as follows: first the visual navigation system was implemented again and the NNEs proposed were tested with different multi-layer perceptron feed-forward network [3] architectures. After that, new voting schemes were implemented and the final recognition results' precision were compared. Finally the best ANN and NNE were re-implemented using fixed-point arithmetic and the results were compared to the results of floating-point version considering precision, execution time, memory consumption and accuracy. In order to verify the hardware consumption of a floating-point unit, nine different Nios II processors were configured on a Altera's Cyclone II FPGA [14] and their resource consumption were compared.

A. *Visual Navigation System*

The visual navigation system described in this subsection is proposed in [11]. This system is partially described in this paper only with respect to the points that are relevant to this work, and any other necessary information about the system can be found in [11]. Visual navigation systems use an image of the environment as an input and the output can be, for example, an image with its pixels classified, a set of actions that will be executed by the robot or a map that is being constructed or corrected. The system proposed in [11] takes an image of the environment divided in blocks of pixels as an input and the output is the same image with the pixels blocks classified as road or non-road pixels. With this information the robot can correct its position to hit only the road pixels on the image, what means that he stays on the road.

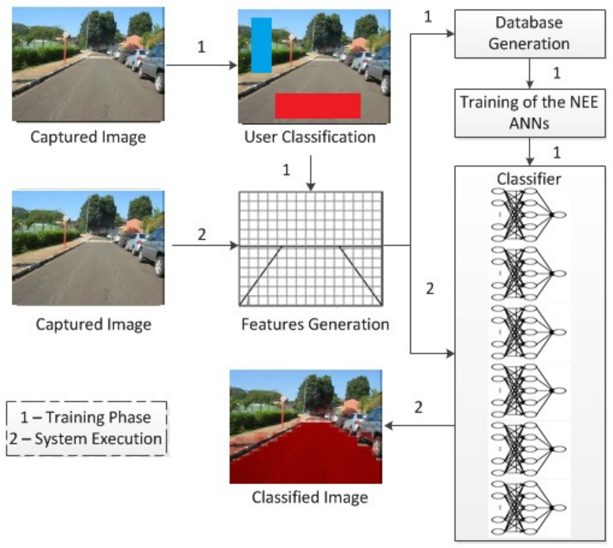


Fig. 1. Classification System Diagram

Figure 1 is a diagram that represents the classification system. First the user, in this case representing the specialist, classifies parts of the image as road pixels (red ones) an parts of the image as non-road pixels (blue ones). After that, the image features for each block of pixels are generated and the database of input-output pairs (that is necessary for the ANNs training algorithm) is created from the classified pixels with the following sets of image features proposed in [11]:

- First Set: U Mean, V Mean, B normalized Mean, H Entropy, G normalized Energy and H Mean.
- Second Set: V Mean, H Entropy, G normalized Energy, G Mean, U Mean, R Mean, H Mean, B normalized Mean, G normalized Mean e Y Entropy.
- Third Set: U Mean, B normalized Mean, V Mean, B Variance, S Mean, H Mean, G normalized Mean, G normalized Entropy.

Each set of image features is used to create the database that was used for a pair of networks, resulting in this case on a classifier (NNE) composed by six ANNs. After training, the system works receiving an image, generating image features and classifying it. The output of the system use grayscale values where the non-road pixels (0) are black, road pixels (1) are white. The intermediate pixels generated by average voting scheme use gray values proportional to the distance to the base pixels.

The activation function of each layer is not described in [11] but considering the possible results, a sigmoid function or a sigmoid symmetric function can be used. In this work two activation functions were considered to the output, and the hidden layers used only sigmoid symmetric function. For training and testing the cross-validation strategy [4] was used considering about 80% of the input-output pairs for training and the other 20% for validation. Training stopping criteria considered 5000 epochs.

One of the important questions about NNEs is how to gather the classifiers outputs into a single output. There are some proposed methods to perform this task and some of them are described in sec. II.B.. Use the mean, as proposed in [11], can be a good choice but in this case the choice was not justified and this fact suggests that other voting schemes should be tested.

B. Voting Schemes for NNEs

Neural Network Ensembles have considerable advantages compared to a single network. Considering that selecting the weights of a Neural Network is an optimization problem with many local minima and that these weights are initialized randomly, different networks make errors on different parts of the input space. As each network makes generalization errors on different subsets of the input space, the decision produced by the ensemble is less susceptible to errors than the decision of an individual network [4]. Another important fact is that, when one network fails the others can be used as well creating a fault-tolerant system.

Despite its advantages, NNEs implementation also has some problems. One of these problems is how to turn the results of a group of networks into a single result as correctly as possible. There are some methods (voting schemes) which were analyzed and investigated to solve this problem and each one achieved better results in specific situations. To choose what methods were going to be tested in this work the future hardware implementation of the system was considered and consequently the mathematically simple methods had advantages. The chosen methods are:

- Accuracy by agreement (Plurality voting): if more networks agree in the classification the chance that the classification is accurate increases [15].
- Average: use the average of the outputs[15].
- Majority voting rule: chooses the classification made by more than half the networks. When there is no agreement among more than half the networks, the result is considered an error[4].

Classification tasks commonly achieved better results with plurality voting or majority voting while simple averaging or weighted averaging are more suitable for regression tasks [12][13]. Therefore these two methods suggested in the literature should be compared to the average method that was chosen in [11].

C. Fixed-Point Implementation

ANNs in this work were implemented using FANN library [16]. In this case the choice is based on two important features of this library: (i) the library is implemented in C language and has a very low execution time for training and use; (ii) the networks can be saved and used after training in fixed-point arithmetic. The second feature is specially interesting for this work. There a many ways to implement fixed-point in ANNs, and a lot o things to consider. This section describes the FANN decisions to fixed-point implementation.

As mentioned in [16] the main questions about the fixed-point implementation are: what functionalities of the library should be implemented in fixed point and what should be done when overflow occurs. The training functionalities of the library were not implemented in fixed point because generally the training phase is done offline by a personal computer with a floating-point unit inside the processor. After training the weights are static and is possible to make one check that will guarantee that an overflow will never occur.

There are several ways to check the weights and guarantee that overflow will not be a problem. The chosen strategy consists of a verification, during the training phase, that will generate the configuration parameters for the fixed-point execution of the network. The decimal point is positioned when the weights of the trained ANN are being saved with fixed-point weights. This procedure will result in a higher precision and less ANNs will fail the check that ensures that an overflow will not occur. To decide the number of bits on the fractional part of the number that are necessary to avoid overflow (t), considering $y = bits(x)$ where y is the number of bits used to represent the integer part of x, the algorithm executes the following calculus for w_i weight and x_i input:

$$t + bits(w_i) + t + bits(x_i) = \quad (1)$$

$$t + bits(w_i) + t + 0 = \quad (2)$$

$$2t + bits(w_i) \quad (3)$$

The number of bits (n_{bits}) needed for the activation function operation is calculated by eq. 4:

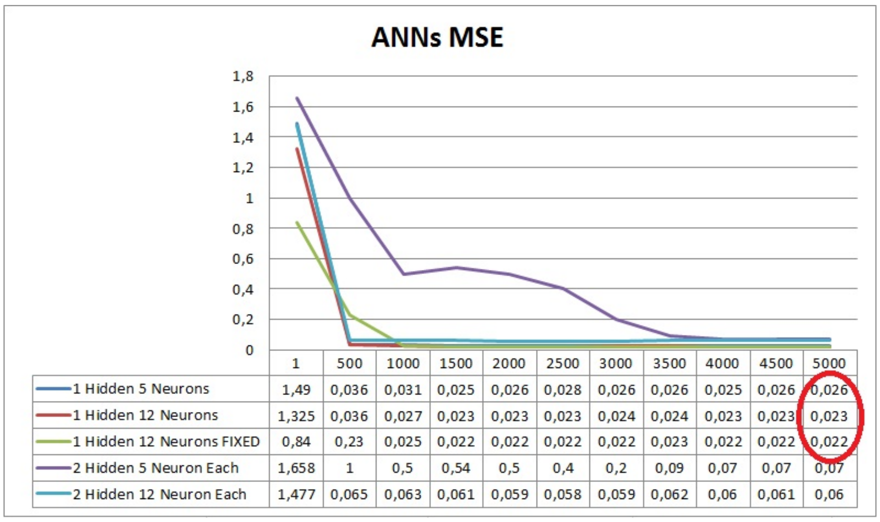
$$n_{bits} = 2t + bits(w_{n+1} \sum_{i=0}^{n} w_i x_i) + 1 \quad (4)$$

Using described strategy the fixed-point network was implemented in FANN library. The consequences of using fixed-point arithmetic are described in the next section together with the other results of this work.

III. RESULTS AND ANALYSIS

The first results of the new implementation are related to ANN architecture. The activation functions had a similar behavior but when sigmoid symmetric is used and the non-road output value is changed from 0 to -1 the MSE during the training phase grows significantly. Consequently the activation function chosen was sigmoid symmetric and the values for road and non-road pixels are 1 and 0 respectively due to test results. The sigmoid activation function also achieved high MSEs.

After choosing the activation function, the networks were implemented and their performance were compared. The training time increases significantly with more than one layer and the MSE doesn't justify the use of more than one layer. With 12 neurons on the hidden layer the networks achieved the lower MSE (fig. 2) and the training time was not significantly increased. This result contrast to presented results from [11]. The number of epochs needed to achieve a



	1	500	1000	1500	2000	2500	3000	3500	4000	4500	5000
1 Hidden 5 Neurons	1,49	0,036	0,031	0,025	0,026	0,028	0,026	0,026	0,025	0,026	0,026
1 Hidden 12 Neurons	1,325	0,036	0,027	0,023	0,023	0,023	0,024	0,024	0,023	0,023	0,023
1 Hidden 12 Neurons FIXED	0,84	0,23	0,025	0,022	0,022	0,022	0,022	0,023	0,022	0,022	0,022
2 Hidden 5 Neuron Each	1,658	1	0,5	0,54	0,5	0,4	0,2	0,09	0,07	0,07	0,07
2 Hidden 12 Neuron Each	1,477	0,065	0,063	0,061	0,059	0,058	0,059	0,062	0,06	0,061	0,06

Fig. 2. Neural Net MSE according to the number of neurons in the hidden layer

acceptable MSE also changed from [11]. With the networks configured in this work only 1500 epochs are required, not 5000 (fig. 2).

Separately analyzing the six networks results for the validation sets, generated with 20% of the initial data available, the results were good as expected. The best precision achieved is 60% of the 12-neuron hidden layer network. It is also important to say that the best results were achieved using the First set of image features (defined in section II.A.). The other two sets of features were not sufficient for the networks alone to classify correctly more than 50% of the images. This results are curious because the networks with 5 neurons on one hidden layer achieved precision of 94% in [11]. These results indicate that some other network parameters were changed and not described or that the FIT parameter [11] is considerably different compared to the MSE considering that the MSE of the best network was 0.023.

In order to improve the ANNs' results accuracy the ensemble was implemented with six networks containing one hidden layer with five neurons. The three voting techniques chosen, and presented in section III.B., were evaluated. The results are shown in table I and, as expected after reading [4][12][13][15], the other techniques achieved a better result than the Average. In case of only two classes, the majority voting and the Accuracy By Agreement achieved the same results.

TABLE I
VOTING TECHNIQUES COMPARISON

Classifier	Accuracy
Network 1 Hidden Layer of 12 neurons	60%
Ensemble Voting Average	40,17%
Ensemble Voting Majority/ABA	65%

After choosing the best networks and the best voting technique for this work's NNE, the networks were evaluated in fixed-point implementation. All networks in this work are fully connected, so the number of weights per network are:

- 35 weights for the First Features Set Networks: 6 inputs *x* 5 hidden neurons + 5 hidden neurons *x* 1 output.
- 55 weights for the Second Features Set Networks: 10 inputs *x* 5 hidden neurons + 5 hidden neurons *x* 1 output.
- 45 weights for the Third Features Set Networks: 8 inputs *x* 5 hidden neurons + 5 hidden neurons *x* 1 output.

The ensemble's total number of weights is 270. If these floating-point weights are stored on 32-bits, the total amount of memory that is necessary to store the ensemble weights is 8640 bits. The fixed-point implementation of FANN stores the same numbers considering two integer parts: one of 14 bits and the other of about 8 bits. This implementation saves 10 bits per weight. This value seems very low but some embedded systems based on microcontrollers have only 2Kb of memory for the entire system. It is also important to remember that if you use only integer numbers, it is not necessary to have a Floating-Point Unit (FPU) that implements the floating-point number operations. To show exactly how much hardware is consumed to implement a floating-point unit, many versions of Nios II soft-processor [14] system were implemented with and without the FPU. It is interesting to notice that the most powerful processor (with all the hardware accelerations) needs 2% more device area only to implement the FPU compared to the same processor without it. Therefore considering the total area and memory saved by fixed-point implementation of ANNs, it is still necessary to verify the fixed-point network precision.

All of the previous developed networks were saved using floating-point and fixed-point. Previous results showed that the best accuracy was achieved by the ensemble using the majority/ABA voting scheme. The six networks of the ensemble were executed in their fixed-point versions to be evaluated and to allow the comparison between the results.

The comparison is shown graphically in figure 3. This graphic represents the number of pixel blocks that composed

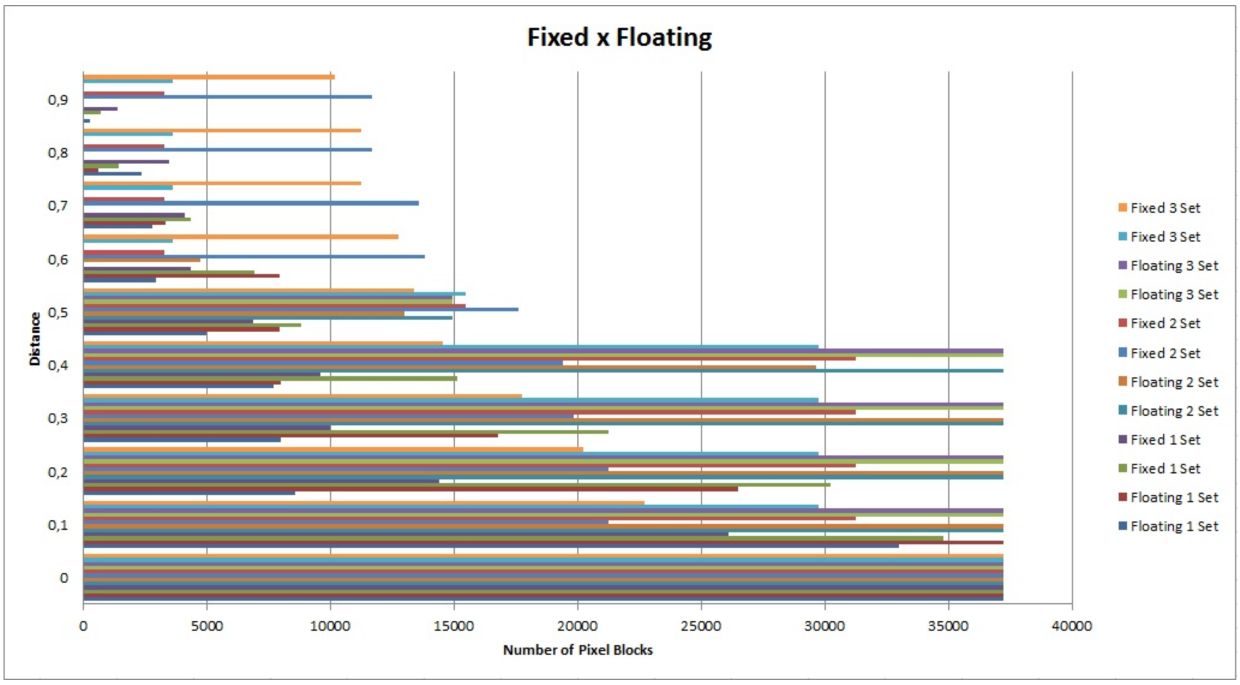


Fig. 3. Fixed and floating-point Comparison

the validation set for cross-validation strategy by the difference between the given answer of the networks and the expected answer. The 12 networks compared were classified by its representation (fixed for fixed-point and floating for floating-point) and by the features set that was used to train the network (defined in section II.A.). Each value of the graphic represents the number of pixel blocks that achieved difference less than the distance value. For example when the difference is 0, that means that the output of the network is exactly the expected output, all values show the maximum number of patterns and that happens because even for the best classifications the output is about 0.9999 and not 1.0 (in this case the difference is 0.0001). As long as the difference grows, the numbers of wrong classified patterns decreases. This graphic is interesting because it shows the results of all ensemble networks for the validation set and it shows that that the results of the fixed-point network are considerably less accurate comparing to the floating-point networks. This is the result of the lower precision of these numbers because of their fixed representation, but this fact alone was not responsible for the worse results.

This bad results of the fixed-point networks were caused by the lack of normalization on the input data. The proposed sets of image features are composed by values that are not normalized between -1 and 1. This fact caused overflows that could not be avoided while the library was converting the network. This problem can be solved by normalizing the input data, but before normalization the ensemble behavior of the fixed-point networks was evaluated. The results are presented in table II.

The results of the ensembles were unacceptable and the average achieved a better result in this case because is less affected by the number of errors than the majority voting or the ABA. The fixed-point networks have a higher error rate that impact directly in the results of the voting, caused by the lack of data normalization. In this case the result was worse because the networks were not so well-converted. This result shows that even with good networks executed in fixed-point the voting schemes for fixed-point network ensembles should be reviewed considering the high error rate. Other procedure that can smooth the impact on voting is to use only networks that were trained with the best features for classification in order to obtain lower number of errors or results that are wrong but not sufficient to have a high influence in the voting schemes.

TABLE II
VOTING TECHNIQUES COMPARISON BETWEEN IMPLEMENTATIONS

Classifier	Accuracy Floating-Point	Accuracy Fixed Point
Network 1 Hidden Layer 12 neurons	60%	35%
Ensemble Voting Average	40,17%	2%
Ensemble Voting Majority/ABA	65%	0%

After the ensemble results, the network that achieved the best accuracy was re-implemented using fixed-point and with data normalization. All the inputs of the network were normalized between -1 and 1. The network was composed by one hidden layer of 12 neurons, 6 inputs and one output. After data normalization the MSE achieved after 1500 epochs were 0.022 2. That was the lower MSE achieved comparing all networks implemented for this work. Also a floating-point network with the same configuration was configured after the same training.

Normalization of the input data changed significantly the results of the networks. The same network using floating-point numbers achieved a accuracy of 70% while the fixed-point network achieved 66% for the same validation set. The difference is very small and the fixed-point network with data normalization achieved a better result than the previous implemented ensemble result of 65%. Considering that the time to do this data normalization is not significant, this procedure can be adopted to achieve better results in this work. The bit need for the representation decreased from 14 to 12.

Other point that is relevant for fixed-point implementations is the execution time. In this case the networks were executed on a Inter Core 2 Duo with 2GB of RAM running CentOS Linux version 5.5. The fixed-point implementation achieved almost half of the floating-point implementation execution time considering the same input. The execution time is very important for the visual navigation system and can be the be the difference between a normal system and a real-time system.

IV. CONCLUSIONS

This work presented the implementation of ANNs and NNEs for a visual navigation system. The visual navigation system was proposed in [11] but some questions about the configuration and implementation of ANNs were not present and consequently some changes had to be done and new techniques were also tested. The fixed-point implementation of the networks was proposed, implemented and analyzed.

The networks implemented in [11] were small compared to the network that achieved the best results in this work. The configuration with one hidden layer of 12 neurons had a fast training phase and the best accuracy. The activation function was also defined and the number of 1500 epochs were sufficient for the network to achieve a training MSE stability.

The ensembles were also implemented. The results showed that the major voting and ABA techniques achieve better results than the average in classification tasks, consequently in this case. The networks with 5 hidden layer when forming a ensemble achieved a 5% of increase on accuracy using the better voting schemes. Considering the fixed-point networks, the ensembles' voting techniques were not sufficient to achieve the desirable accuracy. The voting schemes, when fixed-point networks are used, have to change or at least be reviewed. Also, all the networks should consider the same input features because this procedure can decrease the error by the large exploration of the solution space.

The fixed-point version of the best network achieve only 4% less accuracy compared to the floating-point implementation. The execution time is also faster achieving almost 50% of the floating-point networks' execution time. The way that fixed-point network is implemented on the FANN library achieved good results and can be used for further hardware implementation of the system. The memory necessary to store the weights is considerably lower for the fixed-point network, another important feature for hardware implementation. The accuracy of the networks were low considering the MSE but with the achieved results, that were the main goals of this work, can help with the next experiments and probably the next NNEs will be able to achieve better accuracy considering MSE and the FIT rate [11].

Considering all previous information, for systems that need faster execution times, the fixed-point network using the described implementation should be used. The ensemble techniques achieved better results for floating-point implementations but have to be reviewed for fixed-point networks. The next steps for this work are the definition of new voting schemes and techniques for fixed-point neural network ensembles and the hardware implementation for the visual navigation system to achieve better execution times. Other ANNs also have to be tested to achieve a better MSE.

ACKNOWLEDGMENT

The authors acknowledge the support granted by CNPq and FAPESP to the INCT-SEC (National Institute of Science and Technology – Critical Embedded Systems – Brazil), processes 573963/2008-8 and 08/57870-9. Also acknowledge CAPES for financial support.

REFERENCES

[1] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. springer, 2009.
[2] ——, *The 2005 DARPA Grand Challenge: The Great Robot Race*. springer, 2007.
[3] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.
[4] L. K. Hansen and P. Salamon, "Neural network ensembles," *Transactions on Pattern analysis and Machine Intelligence*, vol. 12, no. 10, 1990.
[5] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, 2010.
[6] M. I. de la Fuente, J. Echanobe, I. del Campo, L. Susperregui, and I. Maurtua, "Hardware implementation of a neural-network recognition module for visual servoing in a mobile robot," in *Proceedings of the 2010 Workshops on Database and Expert Systems Applications*, ser. DEXA '10, 2010, pp. 226–232.
[7] A. Savran and S. nsal, "Hardware implementation of a feed forward neural network using fpgas," *The third International Conference on Electrical and Electronics Engineering ELECO 2003*, vol. 18, no. 3, p. 37, 2003.
[8] D. Dabrowski, E. Jamro, and W. Cioch, "Hardware implementation of artificial neural networks for vibroacoustic signals classification," *ACTA PHYSICA POLONICA A*, vol. 118, no. 1, pp. 41–44, 2010.
[9] B. Jian and Z. Bin, "Optimization of neural network with fixed-point weights and touch-screen calibration," in *Proceedings of ICIEA*, 2009, pp. 226–232.
[10] B. Horne, M. Jamshidi, and N. Vadiee, "Neural networks in robotics: A survey," *Journal of Intelligent and Robotic Systems*, vol. 3, pp. 51–66, 1990.
[11] P. Y. Shinzato, "Sistema de identificacao de superficies navegaveis baseado em visao computacional e rnas," MSc in Computation, ICMC-USP), 2010.
[12] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: Many could be better than all," *Artificial Intelligence*, 2002.
[13] A. Krogh and J. Vedelsby, "Neuralnetworkensembles,cross validation,and activelearning," *Advances in Neural Information Processing Systems*, 1995.
[14] A. Corp., "Altera.com," http://www.altera.com/literature, 2010, acesso: 22/05/2010.
[15] R. Battiti and A. M. Colla, "Democracy in neural networks: Voting schemes for classification," *Neural Networks*, vol. 7, 1994.
[16] S. Lenissen, "Fast artificial neural networks fann," http://leenissen.dk/fann/wp/, 2012, acesso: 20/06/2010.